VSSDB: A Verifiable Secret-Sharing Distance-Bounding Protocol

Sébastien Gambs[1], Marc-Olivier Killijian[2], Cédric Lauradoux[3], Cristina Onete[1], Matthieu Roy[2], Moussa Traoré[2].

Université de Rennes 1 - Inria/IRISA[1],
sebastien.gambs@irisa.fr, cristina.onete@gmail.com,
LAAS-CNRS[2],
{marco.killijian,matthieu.roy,moussa.traore}@laas.fr,
INRIA Rhônes-Alpes[3]
cedric.lauradoux@inria.fr

Abstract. Terrorist fraud is a class of relay attacks against distance-bounding (DB) protocols in which a distant malicious prover colludes with an attacker located in a verifier's proximity when authenticating. Existing DB protocols resisting such attacks are designed to be lightweight and thus symmetric, relying on a secret shared by the prover and the verifier. Recently, several asymmetric distance-bounding protocols were proposed by Gambs, Onete and Robert as well as by Hermans, Peter and Onete, but they fail to thwart terrorist fraud. One earlier asymmetric protocol aiming to be terrorist-fraud resistant is the DBPK-Log protocol due to Bussard and Bagga, which was unfortunately recently proven to achieve neither distance- nor terrorist-fraud resistance. In this work, we build on some ideas of the DBPK-Log scheme and propose a novel DB protocol resistant to terrorist fraud that does not require the pre-existence of a shared secret between the prover and the verifier. Our construction, denoted as VSSDB (for Verifiable Secret Sharing and Distance-Bounding Protocol) relies on a verifiable secret sharing scheme and on the concept of modes, which we introduce as a novel element to complement fast-round challenges in order to improve security. We prove that VSSDB resists mafia-, distance-, and terrorist fraud, as well as impersonation attacks.

1 Introduction

Authentication is a critical building block in cryptography and security, enabling a *prover* to demonstrate his legitimacy to a *verifier*. Whereas most secure authentication protocols provide partial protection against some Man-in-the-Middle (MIM) attacks, they are completely insecure against *relay* attacks. Such threats are critical to authentication and to proximity-testing protocols, as demonstrated for RFID-based Passive Keyless Entry and Start (PKES) systems in cars [16], NFC smartcards [17] and geosocial networks such as Foursquare [10]. A relay adversary takes advantage of the fact that the prover is far away from the verifier

and unaware of the attack. This adversary relays information between the two honest parties in order to impersonate the prover.

Distance-bounding (hereafter denoted DB) protocols were specifically designed by Brands and Chaum [7] to counter relay attacks (also called mafia fraud [11]). They allow the verifier to authenticate the prover only if the latter is within proximity. This proximity testing is done by means of a clock, which the verifier uses to measure the round trip time (RTT) between the sending of a challenge and the reception of the associated response. While DB protocols are generally still only a theoretical concept, they are now also optionally available on a commercial product, the NXP's MIFARE Plus RFID card.

DB protocols must be able to withstand four main types of attacks. The most basic attack is *impersonation* (also called soundness by Hermans, Peeters, and Onete [20]) in which the MiM adversary aims at impersonating a legitimate prover in his absence. A more advanced attack is *mafia fraud*, in which the MiM adversary $\mathcal{A}$ can directly interact with the prover during the impersonation. The main difficulty is that $\mathcal{A}$ cannot simply relay messages during the fast rounds since this will be detected as a delay by the verifier's clock. In *distance fraud*, the prover is malicious and outside proximity (as measured by t_{max}), and he aims to authenticate to the verifier.

Finally, the last of the classical attacks against DB protocols is called *terrorist fraud* [4]. In this attack, a MiM adversary colludes actively with a malicious prover outside his proximity to authenticate (in contrast, in mafia fraud the prover is honest and unaware of the MiM adversary). Terrorist fraud is the most controversial of the four attacks, being also differently formalized in the three main security models in DB literature [1,12,15,26], as we also discuss in Section 2.

Terrorist fraud is a very strong attack and thus many existing DB protocols (notably including that of Hancke and Kuhn [19]) do not address it. In general, most DB protocols in the literature are lightweight and *symmetric*. This is the case for the Swiss-Knife protocol [21] and for the scheme of Avoine, Lauradoux, and Martin [2], which address terrorist fraud by using secret sharing. In these schemes, the shared private key is masked by another random string and used to respond to fast challenges. The class of SKI protocols [6], which prevent terrorist fraud in a provable way, is also symmetric. A notable exception to this approach is the DBPK-Log protocol [9] due to Bussard and Bagga. Currently, none of the *asymmetric* (public-key) DB protocols existing in the literature [7,20,18] prevent terrorist fraud. In addition, recent analyses show that the DBPK-Log and other protocols [2,3,14,27] aiming for terrorist-fraud resistance, are vulnerable to attacks such as a form of key leakage.

Our contributions. To address this issue, we propose an asymmetric *Verifiable Secret Sharing and Distance-Bounding* protocol (denoted Vssdb) that resists all main classes of attacks. One of the novelties of our protocol is the use of *modes*, which determine together with fast-round challenges, the fast-round responses. By using homomorphic commitment schemes and verifying in a bitwise manner the soundness of the shares with respect to the secret key, we prevent the attacks suggested by [3].

Table 1 puts our results in perspective, comparing the properties of our VSSDB protocol to others in the literature. These figures also take into account recent attacks by [3,6,14].

	Protocol \ Fraud	Impersonation	Mafia	Terrorist	Distance
Symmetric	Reid et al. [24]	1	?*	No	?*
	Tu & Piramuthu [25]	1	1	No	$\left(\frac{3}{4}\right)^m$
	Swiss-Knife [21]	$\left(\frac{1}{2}\right)^m + \mathbf{Adv}^{\mathsf{Unf}}_{\mathcal{A}}$	$\left(\frac{1}{2}\right)^m$	No	$\left(\frac{3}{4}\right)^m$
	Avoine and co-authors [2]	1	$\left(\frac{2}{3}\right)^m$	Yes	?**
	$SKI_{\mathbf{pro}}$ [5]	$(\frac{1}{2})^m$	$\left(\frac{2}{3}\right)^m$	Yes	$\left(\frac{3}{4}\right)^m$
Asymmetric	Brands and Chaum [7]	$\left(\frac{1}{2}\right)^m + \mathbf{Adv}^{\mathsf{Unf}}_{\mathcal{A}}$	$\left(\frac{1}{2}\right)^m + \mathbf{Adv}^{\mathsf{Unf}}_{\mathcal{A}}$	No	$\left(\frac{1}{2}\right)^m$
	GOR [18]	$2^{-\lvert\mathbb{G}\rvert}$	$\left(\frac{1}{2}\right)^m$	No	$\left(\frac{3}{4}\right)^m$
	DBPK-Log [9]	$\left(\frac{1}{2}\right)^{4m'}$	$\left(\frac{1}{2}\right)^m$	No	1
	VSSDB	$2^{-2m} + \mathbf{Adv}^{\mathsf{Unf}}_{\mathcal{A}}$	$\left(\frac{1}{2}\right)^m$	Yes	$\left(\frac{3}{4}\right)^m$

Table 1: Security bounds for several DB protocols run with m time-critical rounds. We denote by $\mathbf{Adv}^{\mathsf{Unf}}_{\mathcal{A}}$ the unforgeability bound of a signature scheme. * No generic bounds are applicable. A specific instantiation is NOT mafia-fraud resistant, but resists distance fraud with probability $\frac{3}{4}$ per round [14]. ** In fact, stronger assumptions on f are needed for the claimed bound of $\frac{3}{4}$ per round [3]. The value $m' := 50$ is a system parameter defined by [9].

STRUCTURE. First in Section 2, we provide a brief overview of DB protocols and their security as well as the building blocks such used in our protocol. Afterwards in Section 3, we introduce our VSSDB protocol and state its exact security properties before concluding in Section 4.

2 Preliminaries

2.1 Security of Distance-Bounding Protocols

The principle of DB protocols is that fast (i.e., speed of light) exchanges of bits between a prover P and a verifier V can be timed and used by the verifier to upper-bound his distance to P. By using a clock, V will measure the round-trip time (RTT) between the moment he sends a challenge and the time at which he receives the response. The verifier associates the trusted proximity between himself and the prover with a clock value t_{max}. If the measured RTT value is less than t_{max} and the responses are correct, V accepts P as legitimate while otherwise, the prover is rejected. DB protocols usually consist of multiple

communication phases, also called rounds. The rounds in which the verifier uses his clock to measure the RTT are called time-critical (or fast) while the ones in which the verifier does *not* measure the RTT value are called slow (or lazy).

The security of such schemes was first consistently defined by Avoine and co-authors [1]. Their semi-formal definitions served as a cornerstone to the later formalizations of Dürholz, Fischlin, Kasper, and Onete [12,15], and Boureanu and co-authors [6,26]. While the model of Vaudenay [26] is more generic, as it allows for multiple provers and takes into account distance hijacking, we prefer to use the more practical game-based notion of terrorist-fraud resistance due to Fischlin and Onete [15]. Thus, our proofs are written in their underlying framework [12], which we refer to as the DFKO model.

In this model, the adversary (denoted $\mathcal{A}$) can open *sessions*, to observe honest protocol runs between P and V (such sessions are called *prover*-verifier), or to interact with P (called *adversary-prover*) or with V (called *verifier-adversary*). In the security definitions of each of the attacks, the attack is quantified in terms of the *number* of sessions of each type that he opens. These parameters are denoted, respectively, q_{OBS} for the prover-verifier sessions, q_P for adversary-prover sessions, and q_V for verifier-adversary sessions.

The implicit assumption on DB protocol is that the clock used always correctly detects the fact that the source of a message is outside the verifier's proximity. The two direct consequences are (1) that no MIM adversary can relay the exact same messages between an honest verifier and an honest, but distant prover, and (2) that not even a malicious prover can make his messages travel faster than the communication speed. In particular, such a prover must send the responses *before* receiving the challenges during the time-critical rounds if the responses are to arrive on time. In the DFKO model, the prover must commit to each time-critical response *before* the corresponding round starts. Note that in this case the commitment is not cryptographic – it mainly ensures that the prover chooses the response before receiving the challenge.

We intuitively state below the attacks covered by this formalization [12,15] and refer the interested reader to the original papers for their precise definitions.

Distance-fraud resistance. A prover situated outside the proximity of the verifier should be rejected except with negligible probability.

Mafia-fraud resistance. A MiM adversary, having access to an honest but distant prover and to the verifier, should be rejected except with negligible probability. The verifier's clock is assumed to detect any pure relays by the adversary during the fast rounds.

Slow impersonation resistance. A MiM adversary that can communicate with both the prover and the verifier, but not duplicate the session transcripts, should only pass the slow rounds of the protocol with negligible probability.

Terrorist-fraud resistance. If a malicious prover can help an adversary authenticate to the verifier, this adversary should later have a probability of winning in mafia fraud attacks better than without the prover's help.

The last of these four properties, namely terrorist-fraud resistance, is a controversial notion in the distance-bounding literature. The crux of the definition

is to formalize the restriction on the prover's collusion with the adversary. In particular, this condition is modeled differently in the three existing security models (and their extensions)[2,1,12,15,26]. In particular, Avoine and co-authors require that after the attack, the adversary should have zero-knowledge of the secret. Originally, the DFKO notion of terrorist-fraud resistance [12] was simulation-based and required that if the prover helps a specific adversary $\mathcal{A}$ authenticate with a certain probability, a simulator having access to $\mathcal{A}$'s view should succeed to authenticate in a fresh verifier-adversary session *with the same probability*. This definition was relaxed by Fischlin and Onete [15] to the notion of GameTF-security. This notion demands that after the prover helps a certain adversary $\mathcal{A}$ authenticate with non-negligible probability, this same adversary can then authenticate to the verifier with better than mafia-fraud probability.

2.2 Building Blocks

Hereafter, we briefly highlight the building blocks that we used to construct our solution and provide some intuition about how we use them.

Homomorphic commitments. In our protocol, the prover will commit to each bit of the prover's secret value x using a homomorphic bit commitment scheme. These commitments prevent the direct knowledge of x while ensuring that the prover's responses are tied to the committed bits of the secret key. We overcome the weakness of DBPK-Log through a bitwise verification of x.

Verifiable secret sharing. The use of verifiable secret sharing in DB protocols has already been used by Avoine, Lauradoux, and Martin [2] to thwart terrorist fraud. More precisely, their approach relies on threshold secret-sharing. Their main idea is that if a terrorist adversary is aided by a malicious prover, after k such sessions the adversary will have learned a number of the bits of the secret key, enabling him to later authenticate on his own. In most DB protocols, the prover masks the secret value with a single share. In contrast, we rely on two additional shares in our protocol, which are committed to in the first message of the protocol. These shares serve to construct the fast-round responses, but they always mask x. We refer the interested reader to [13] for the description of Feldman's verifiable secret sharing scheme, which combines classical secret-sharing and homomorphic commitments.

Modes. While the prover keeps x secret from the verifier, we prevent distance fraud by introducing different response formats. These response formats depend on binary challenges *and* on a sequence of *modes*, chosen at random by the verifier among four possible ones. The prover's responses, depending on both modes and challenges, are then verified by opening the corresponding commitments.

Signature of the transcript. In VSSDB, we prevent key-leakage attacks by using the countermeasure of Brands and Chaum [7] and the Swiss-Knife protocol [21], which consists in signing the entire transcript at the end.

3 Verifiable Secret-Sharing DB

We present our protocol VSSDB, in two steps. First, we build on a preliminary construction, which is mafia-, distance-, and impersonation resistant, but does not achieve terrorist-fraud resistance (Section 3.1). Then, we add a backdoor, enabling the terrorist-fraud adversary to use the information learnt from the prover to then authenticate with higher probability to the verifier (Section 3.2).

3.1 The VSSDB protocol

INTUITION. In our protocol, the prover chooses at each session two m-bit long strings – denoted respectively k and ℓ – that are used to mask the private key x, bit by bit, into a new string e. At initialization, the prover commits to these values and sends the commitments to the verifier. The latter generates an m-bit random value (essentially a string of bits $\mathsf{M}_i \in \{0,1\}$) encoding a specific *response mode* among two possible ones. The response bits required from the prover during the time-critical rounds will depend both on the fast-phase challenges *and* on the mode. In particular, the response function, denoted f, takes as input M_i and the round challenge c_i, and outputs a corresponding response r_i. The fact that the modes are chosen honestly by the verifier and the response components (i.e., k, ℓ and e) are committed before receiving the mode values enforce distance-fraud resistance. We hide the committed values from the adversary by encrypting them with the verifier's public key. Furthermore, we prevent the adversary from forwarding self-generated values by including a signature on these values within the plaintext to be encrypted. Finally, we avoid replay attacks by making the verifier send a session-specific nonce at each session.

Given the commitments a_i, b_i, d_i for respectively k_i, ℓ_i, and e_i, the use of homomorphic commitments allows one to verify the relation that $a_i b_i d_i = \mathsf{Com}_i$ (for $i = 1, \ldots, m$). Furthermore, a non-interactive zero-knowledge (NIZK) proof of knowledge ensures that the value used by the prover is really his secret key, consistent with the values Com_i and with the ciphertext $\mathsf{c_P}$. During the time-critical rounds, the verifier measures and stores the RTT of each exchange, denoted δt_i. In order to enable the verifier to check each response value r_i, the prover must then open the relevant commitments of the values used to generate r_i. Thus, in a subsequent verification step, the prover sends the necessary auxiliary information (the randomness used to commit and possibly the response values themselves), which we denote as a function γ, taking two inputs M_i and c_i. Finally, we use the countermeasure proposed in [21] and sign the modes, challenges, and responses used in this session, as well as the session-specific nonce.

In more details, the protocol consists of the following phases.

SET UP. All the provers and verifiers are initialized by a trusted third party (TTP) acting as a registration authority. The TTP begins by setting-up the global parameters. First, it picks at random two distinct large prime numbers p and q and computes their product $n \leftarrow pq$. The TTP also generates $t \xleftarrow{\$} \mathbb{Z}_n^*$

Protocol 1: Verifiable Secret Sharing and Distance-Bounding (VSSDB).

Verifier V		**Prover** P
Public key $\mathsf{pk} = \{\mathsf{Com}, \mathsf{pk}_{\mathsf{Sign}}\}$ with $\mathsf{Com} := (\mathsf{Com}_1, \ldots, \mathsf{Com}_m)$ Private encryption key: sk_{V}		Private key $x, \mathsf{sk}_{\mathsf{Sign}}, \{\nu_1, \ldots \nu_m\}$
	INITIALIZATION PHASE	
		For $i = 1$ to m generate: $k_i, \ell_i \xleftarrow{\$} \{0,1\}$; $e_i = \mathcal{E}_{k_i \| \ell_i}(x_i)$; $u_i, v_i \xleftarrow{\$} \mathbb{Z}_n^*$; $w_i = u_i^{-1} v_i^{-1} \mathsf{H}^i(x) \mod n$; $a_i = \mathsf{Commit}(k_i, u_i)$; $b_i = \mathsf{Commit}(\ell_i, v_i)$; $d_i = \mathsf{Commit}(e_i, w_i)$
	$\xrightarrow{0 \| N_{\mathsf{V}}}$	
		Set: $m_{\mathsf{P}} := \{a_i, b_i, d_i\}_{i=1}^m \| N_{\mathsf{V}}$; Compute: $\mathsf{c}_{\mathsf{P}} = \mathsf{Enc}(\mathsf{pk}_{\mathsf{V}}, m_{\mathsf{P}} \| \mathsf{Sign}(\mathsf{sk}_{\mathsf{Sign}}, m_{\mathsf{P}}))$; Set: $\mathcal{W} := \{x, \{k_i, \ell_i, e_i, u_i, v_i, w_i, a_i, b_i, d_i, \nu_i\}_{i=1}^m, \mathsf{sk}_{\mathsf{Sign}}\}$; $\pi := \mathsf{Prove}\{\mathcal{W} : \mathsf{c}_{\mathsf{P}}$ well formed $\wedge$ consistent with $\{\mathsf{Com}_i\}_{i=1}^m\}$
	$\xleftarrow{\mathsf{c}_{\mathsf{P}}, \pi}$	
Decrypt c_{P} to $m' \| N_{\mathsf{V}}^* \| \sigma^*$; If π and σ^* do not check or $N_{\mathsf{V}}^* \neq N_{\mathsf{V}}$, abort **For** $i = 1$ to m generate $\mathsf{M}_i \xleftarrow{\$} \{0,1\}$		
	$\xrightarrow{\{\mathsf{M}_i\}_{i=1}^m}$	
		Parse as: $\hat{\mathsf{M}}_1, \ldots \hat{\mathsf{M}}_m \in \{0,1\}$; For $i = 1$ to m, compute $\mathsf{f}(\hat{\mathsf{M}}_i, 0), \mathsf{f}(\hat{\mathsf{M}}_i, 1)$, $\gamma(\hat{\mathsf{M}}_i, 0)$, and $\gamma(\hat{\mathsf{M}}_i, 0)$ as below: $\mathsf{f}(0,0) = e_i$; $\mathsf{f}(0,1) = k_i \oplus \ell_i$; $\mathsf{f}(1,0) = k_i$; $\mathsf{f}(1,1) = e_i \oplus \ell_i$; $\gamma(0,0) = w_i$; $\gamma(0,1) = (k_i, u_i, \ell_i, v_i)$; $\gamma(1,0) = u_i$; $\gamma(1,1) = (e_i, w_i, \ell_i, v_i)$
	INTERACTIVE PHASE	
		For $i = 1$ to m
Picks $c_i \xleftarrow{\$} \{0,1\}$	$\xrightarrow{c_i}$	Parse as $\hat{c}_i$
Store r_i and RTT δt_i	$\xleftarrow{r_i}$	$r_i \leftarrow \mathsf{f}(\hat{\mathsf{M}}_i, \hat{c}_i)$
	VERIFICATION PHASE	
	$\xleftarrow{\{\gamma(\hat{\mathsf{M}}_i, c_i)\}_{i=1}^m}$	$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathsf{Sign}}, \{\mathsf{M}_i\}_{i=1}^m \| \{c_i, r_i\}_{i=1}^m \| N_{\mathsf{V}})$
	$\xleftarrow{\sigma}$	

Verify σ, open commit for $\mathsf{f}(\hat{\mathsf{M}}_i, \hat{c}_i)$ with $\gamma(\hat{\mathsf{M}}_i, \hat{c}_i)$, check $\delta t_i \leq t_{\mathsf{max}}\ \forall i$. Accept only if all checks succeed.

such that $t \neq \pm 1$ of order $\varphi(n)/4$ and computes $s = t^2 \mod n$. In particular, $s = -1 \mod n$ and n is such that the Jacobi symbol $(\frac{-1}{n})$ is $+1$. Then, for any two bits a and b, it holds that $s^{a+b} = s^{a \oplus b}$. The parameters n and s are considered to be public and thus known to all parties, while p, q and t are secret parameters known only to the TTP. These parameters will enable the use of the (homomorphic) commitment scheme (originally called a blob) due to Brassard, Chaum, and Crépeau [8].

Later, at prover registration, the TTP generates a private/public signature keypair $(\mathsf{sk}_{\mathsf{Sign}}, \mathsf{pk}_{\mathsf{Sign}}) \leftarrow \mathsf{SSKGen}()$. The signature scheme is modeled as a tuple $\mathcal{S} = (\mathsf{SSKGen}, \mathsf{Sign}, \mathsf{Vf})$ and can be any unforgeable signature scheme (e.g., Elgamal or DSA). We assume that the verifier has certified secret/public keys for an IND-CCA public-key encryption scheme $(\mathsf{EKGen}, \mathsf{Enc}, \mathsf{Dec})$. The TTP is also assume to honestly generate a secret key $x \in \{0,1\}^m$ to each prover such that each bit of the key is 0 with probability 1/2 (independently). This assumption is used in the bound for distance-fraud resistance. Afterwards, each bit x_i is committed to a value Com_i as follows: the TTP generates a witness ν_i and sets $\mathsf{Com}_i \leftarrow \mathsf{z}_i^2 s^{x_i} \mod n$ with $\nu_i = \mathsf{H}^i(x)$. In this equation, H denotes a cryptographically secure hash function with pseudo-random output, while $\mathsf{H}^i(x)$ is the i^{th} iteration of H on x. The values ν_i for each i are given to the prover while the tuple $\mathsf{Com} := (\mathsf{Com}_1, \ldots, \mathsf{Com}_m), \mathsf{pk}_{\mathsf{Sign}}$ is the prover's public key given to all verifiers.

Initialization. As described previously, the verifier generates a session-specific nonce N_{V} that he sends to the prover. We add at the beginning a 0 to this value as an encoding, to differentiate between an honest protocol run and the all-or-nothing disclosure function in Protocol 2. In return, the prover generates randomly two m-long bitstrings k and ℓ. These strings are used to symmetrically encrypt (we specifically use bitwise XOR) each bit x_i of the private key. In particular, $e_i \leftarrow \mathcal{E}_{k_i||\ell_i}(xi) = x_i \oplus k_i \oplus \ell_i$. Subsequently, the prover uses a secure bit-commitment scheme $\mathsf{Cmt} = (\mathsf{Commit}, \mathsf{COpen})$ to commit to each bit k_i, ℓ_i, and e_i, using respectively the randomly generated witnesses u_i, v_i, and $w_i = (u_i v_i)^{-1} \mathsf{H}^i(x) \mod n$. Each commitment takes two inputs, the committed value b and the randomness r, and outputs $\mathsf{Commit}(b, r) := r^2 s^b$ for the parameter s generated at setup. The prover concatenates these commitments and the nonce N_{V}, forming the message m_{P}. This message is signed as $\mathsf{Sign}(\mathsf{sk}_{\mathsf{Sign}}, m_{\mathsf{P}})$ and this signature is appended to the message itself before being encrypted into $\mathsf{c}_{\mathsf{P}} := \mathsf{Enc}(m_{\mathsf{P}}, \mathsf{Sign}(\mathsf{sk}_{\mathsf{Sign}}, m_{\mathsf{P}}))$. Finally, the witnesses consisting of the prover's secret keys x and $\mathsf{sk}_{\mathsf{Sign}}$, the commitments $\{\nu_i\}_{i=1}^m$, as well as the auxiliary values $\{k_i, \ell_i, e_i\}$, the commitment witnesses u_i, v_i, w_i, and the commitment values a_i, b_i, d_i for each i, are used as inputs to a NIZK proof of knowledge π.This NIZK demonstrates that the value x is the one used in computing e_i from k_i and ℓ_i, and that it is also the value used in computing the verifier's public commitments Com_i for all i. Furthermore, the NIZK proof also shows that the ciphertext and signature included in c_{P} are well formed with respect to these values.

The prover sends the ciphertext c_{P} and the NIZK proof π to the verifier, who, upon successful verification of the proof, decrypts the ciphertext and checks

the signature. If this verification is successful, the verifier generates randomly a sequence of m response modes $\mathsf{M}_i \in \{0,1\}$. These modes form one of the two inputs required for the computation of the responses during the time-critical rounds. Finally, the prover pre-computes the responses for each received mode M_i and for each possible bit challenge $c_i = \{0,1\}$. The response function $\mathsf{f} : \{0,1\} \times \{0,1\} \longrightarrow \{0,1\}$ is computed as follows:

$$\mathsf{f}(\mathsf{M}_i = 0, c_i) = \begin{cases} e_i \text{ if } c_i = 0 \\ k_i \oplus \ell_i \text{ if } c_i = 1 \end{cases} \qquad \mathsf{f}(\mathsf{M}_i = 1, c_i) = \begin{cases} k_i \text{ if } c_i = 0 \\ \overline{e_i \oplus \ell_i} \text{ if } c_i = 1 \end{cases}$$

Furthermore, for each tuple of inputs (M_i, c_i), the function $\gamma : \{0,1\} \times \{0,1\} \longrightarrow \mathbb{Z}^{2|u_1|} \times \mathbb{Z}^{2|a_1|}$ lists values allowing the verifier to later open the relevant commitments amongst a_i, b_i, and d_i and verify the responses. Although we define the range of γ to be a tuple of four values, two of the bit-length of the randomness to commit (i.e., u_i) and two of the bit-length of the output commitments (i.e., a_i), the prover sometimes needs to send only a *subset* of these values. In this case by convention the remaining output values are ignored by the verifier. More precisely for each input (M_i, c_i), the function γ outputs the values of the shares used to compute $\mathsf{f}(\mathsf{M}_i, c_i)$ if these are not given in clear (but rather XORed) and the relevant witnesses. For example, since $\mathsf{f}(0,0) = e_i$, the corresponding $\gamma(0,0)$ is equal to w_i, which is the randomness used to construct the commitment d_i of e_i. In contrast, $\mathsf{f}(0,1) = k_i \oplus \ell_i$, thus the output $\gamma(0,1)$ will consist of the two component shares k_i, ℓ_i, as well as the randomness u_i, v_i that opens the commitments a_i and b_i. The complete output space of γ is described in details in Protocol 1.

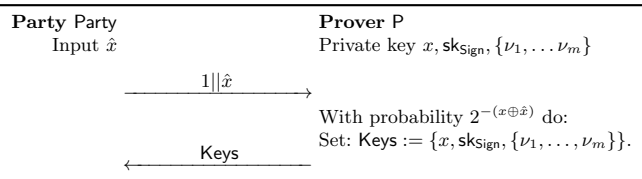
Interactions. The prover and verifier then begin the time-critical rounds, which are effectively a sequence of m challenge-response exchanges in which the verifier sends a bit challenge c_i and the prover replies with the response bit $\mathsf{f}(\mathsf{M}_i, c_i)$. The verifier stores the RTT of the exchange, which we denote as δt_i. We point out that implementing a precise measurement of the RTT during the challenge-response rounds is still a difficult technological challenge, although one implementation has recently been proposed [23].

Verification. During this phase, the prover provides the output $\gamma(\mathsf{M}_i, c_i)$ for each round, as well as a final signature, generated with $\mathsf{sk}_{\mathsf{Sign}}$, of the modes, the concatenation of challenges and responses, and the session nonce N_{V}. Then, the verifier performs the following verification steps: (1) he checks the received signature $\hat{\sigma}$ based on his view of the transcript, (2) he opens the commitments and validates the received responses, (3) he checks that $\delta t_i \leq t_{\mathsf{max}}$ for all $i = 1$ to m, in which t_{max} is the trusted proximity bound, and (4) he checks that for each $i = 1$ to m, it holds that $\mathsf{Com}_i = a_i b_i d_i \mod n$. If any of these verification steps fails, the verifier aborts, which result in rejecting the prover. Otherwise, the verifier accepts (i.e., authenticates) the prover.

3.2 Introducing Cheat Modes

In order to ensure terrorist-fraud resistance, we have to build a backdoor into the protocol. To do this, we use an idea similar to that proposed by Fischlin and Onete [15]. However, we cannot directly import their solution as our protocol is asymmetric, whereas the original trick in [15] assumes that the verifier knows the prover's secret key. Instead, we take an approach similar to all-or-nothing security and modify the protocol in two steps. More precisely during the first step, if the prover receives a message starting with a 1 instead of the first protocol message $(0|N_{\mathsf{V}})$, he will compare the remainder of the message with the secret key x. If the received message is sufficiently close, he returns the full key x as well as the witnesses ν_i that were used by the TTP to compute the commitments Com_i as depicted in Protocol 2.

Protocol 2: All-or-nothing key disclosure.



Any party, including an adversary, can run this strategy. However, note that the protocol does *not* return the secret signature key of the prover, thus on its own knowing x will not significantly help an adversary to authenticate due to the final signature. To enable a successful terrorist-fraud adversary to authenticate afterwards without the prover's help, we also change the authentication protocol. More precisely, we introduce a "cheat" running mode, in which a party can use the values of x and ν_i to authenticate just by echoing the received challenges. As an honest prover only discloses the values of x and ν_i in exchange for a good guess of the secret key x, these values will not occur when observing honest-prover to honest-verifier communications. We also prevent distance fraud, as using this strategy will reduce the success probability of a distance-fraud adversary to just $\frac{1}{2}$ per round (the probability of guessing the challenges in advance).

However, if a malicious prover helps a MIM adversary to authenticate during terrorist fraud, the adversary will learn a significant part of the secret key x. This will enable the terrorist-fraud MIM adversary to learn (with high probability) the values of x and ν_i, which in turn enable him to make the verifier run the protocol in cheat mode, as depicted in Protocol 3.

Protocol 3: Terrorist VSSDB.

Verifier V		**Prover** P
Public key $\{x, \mathsf{Com}, \mathsf{pk}_{\mathsf{Sign}}\}$		Private key $x, \mathsf{sk}_{\mathsf{Sign}}$
with $\mathsf{Com} := (\mathsf{Com}_1, \ldots, \mathsf{Com}_m)$		
Private encryption key: sk_{V}		
	INITIALIZATION PHASE	
	$\xrightarrow{0\|\|N_{\mathsf{V}}}$	Initialize as in Protocol 1
		If first bit is 0, run Protocol 1
	$\xleftarrow{0\|\mathsf{c}_{\mathsf{P}}, \pi}$	Else react as in Protocol 2 and halt
If first bit is 0, run Protocol 1		
Else: parse message as $\hat{x}, \{\hat{\nu}_i\}_{i=1}^m$		
If $\exists i$ such that $\mathsf{Commit}(\hat{x}_i, \hat{\nu}_i) \neq \mathsf{Com}_i$ reject P		
Else, set $\mathsf{SBit} = \mathtt{cheat}$		
	INTERACTIVE PHASE	
	For $i = 1 \cdots m$	
Pick $c_i \xleftarrow{\$} \{0,1\}$	$\xrightarrow{c_i}$	
Store r_i and RTT δt_i	$\xleftarrow{r_i}$	
	VERIFICATION PHASE	
Accept if and only if for all i, $r_i = c_i$ and $\delta t_i \leq t_{\mathsf{max}}$		

3.3 Security Analysis

Despite the fact that our protocol is presented in two steps, we provide below the security statement corresponding to the full construction (including the cheat modes). We refer the reader to Appendix for the detailed proofs.

Theorem 1. *The* VSSDB *protocol (including the cheat modes described in Protocols 2 and 3) has the following properties, if the function* H *produces pseudorandom output:*

- *For any* (t, q_V)*-distance-fraud adversary* $\mathcal{A}$ *against VSSDB, there exist the following adversaries:* $\mathcal{A}_1$ *against the pseudorandomness of the output of* H*,* $\mathcal{A}_2$ *against the binding property of the commitment scheme* $\mathsf{Cmt} = (\mathsf{Commit}, \mathsf{COpen})$*,* $\mathcal{A}_3$ *against the correctness of the symmetric encryption scheme,* $\mathcal{A}_4$ *against the correctness of the commitment scheme* Cmt*,* $\mathcal{A}_5$ *against the soundness of the NIZK-PK proof* π*, and* $\mathcal{A}_6$ *against the binding property of the commitment scheme such that:*

$$\begin{aligned}\boldsymbol{Adv}_{\mathcal{A}}^{\mathsf{Dist}} \leq\ & mq_V \boldsymbol{Adv}_{\mathcal{A}_1}^{\mathsf{H-PRF}} + 3mq_V \boldsymbol{Adv}_{\mathcal{A}_2}^{\mathsf{Com.Bind}} + q_V m \boldsymbol{Adv}_{\mathcal{A}_3}^{\mathcal{E}().\mathsf{Corr}} \\ & + q_V m \boldsymbol{Adv}_{\mathcal{A}_4}^{\mathsf{Com.Corr}} + q_V \boldsymbol{Adv}_{\mathcal{A}_5}^{\mathsf{NIZK.Sound}} + mq_V \boldsymbol{Adv}_{\mathcal{A}_6}^{\mathsf{Com.Bind}} + q_V \left(\frac{3}{4}\right)^m.\end{aligned}$$

- *For any* $(t, q_{\mathrm{OBS}}, q_P, q_V)$*-mafia-fraud adversary* $\mathcal{A}$ *against* VSSDB*, there exist the following adversaries:* $\mathcal{A}_1$ *against the correctness of the public-key*

encryption scheme, $\mathcal{A}_2$ against the correctness of the signature scheme $\mathcal{S} = (\mathsf{SSKGen}, \mathsf{Sign}, \mathsf{Vf})$, $\mathcal{A}_3$ against the unforgeability of the same signature scheme, $\mathcal{A}_4$ against the IND-CCA security of the public-key encryption scheme $(\mathsf{EKGen}, \mathsf{Enc}, \mathsf{Dec})$, and $\mathcal{A}_5$ against the zero-knowledge property of the NIZK, such that:

$$\begin{aligned}\boldsymbol{Adv}_{\mathcal{A}}^{\mathsf{Mafia}} \leq\ & q_V \left(\frac{1}{2}\right)^m + q_V\, \boldsymbol{Adv}_{\mathcal{A}_1}^{\mathsf{PKE.Corr}} + q_V\, \boldsymbol{Adv}_{\mathcal{A}_2}^{\mathsf{Sign.Corr}} + q_V\, \boldsymbol{Adv}_{\mathcal{A}_3}^{\mathsf{Unf}} \\ & + (q_{\mathrm{OBS}} + q_P)(\boldsymbol{Adv}_{\mathcal{A}_4}^{\mathsf{PKE.IND-CCA}} + \boldsymbol{Adv}_{\mathcal{A}_5}^{\mathsf{NIZK.ZK}}) + q_V 2^{-m} \\ & + q_P 2^{(-2-\log_2 3)m} + \binom{q_V}{2} 2^{-|N_{\mathsf{V}}|}.\end{aligned}$$

– *For any $(t, q_{\mathrm{OBS}}, q_P, q_V)$-impersonation adversary $\mathcal{A}$ against* VSSDB*, there exist the following adversaries: $\mathcal{A}_1$ against the correctness of the public-key encryption scheme, $\mathcal{A}_2$ against the correctness of the signature scheme $\mathcal{S} = (\mathsf{SSKGen}, \mathsf{Sign}, \mathsf{Vf})$, $\mathcal{A}_3$ against the unforgeability of the same signature scheme, $\mathcal{A}_4$ against the IND-CCA security of the public-key encryption scheme $(\mathsf{EKGen}, \mathsf{Enc}, \mathsf{Dec})$, and $\mathcal{A}_5$ against the zero-knowledge property of the NIZK, such that:*

$$\begin{aligned}\boldsymbol{Adv}_{\mathcal{A}}^{\mathsf{ImpSec}} \leq\ & q_V\, \boldsymbol{Adv}_{\mathcal{A}_1}^{\mathsf{PKE.Corr}} + q_V\, \boldsymbol{Adv}_{\mathcal{A}_2}^{\mathsf{Sign.Corr}} + q_V\, \boldsymbol{Adv}_{\mathcal{A}_3}^{\mathsf{Unf}} + \binom{q_V}{2} 2^{-|N_{\mathsf{V}}|} \\ & + (q_{\mathrm{OBS}} + q_P)(\boldsymbol{Adv}_{\mathcal{A}_4}^{\mathsf{PKE.IND-CCA}} + \boldsymbol{Adv}_{\mathcal{A}_5}^{\mathsf{NIZK.ZK}}) + + q_P 2^{(-2-\log_2 3)m}.\end{aligned}$$

– *The* VSSDB *protocol is GameTF-secure.*

4 Conclusion

In this paper, our main contribution is a DB protocol called VSSDB, which is asymmetric (i.e., the prover and verifier do not share a common secret) while provably resisting terrorist-fraud attacks (as captured by the GameTF flavor of terrorist-fraud resistance [15]). One of the main novelties of our protocol is the use of *modes*, which are communicated during slow phases, and complement the challenges sent during the time-critical rounds. More precisely, the responses answered by the prover during one of these rounds depend both on the mode of this round *and* the challenge. This feature improves the protocol's security, while preserving the lightweightness of computations performed during the time-critical rounds. In addition, in contrast to other DB protocols, we rely on the use of *three* shares rather than two, which allows for more richness in the responses while better hiding the secret. In particular, by using a homomorphic commitment scheme and *bitwise* verification of the commitment, we essentially prevent the attack of Bay and co-authors [3] against DBPK-Log scheme on which VSSDB builds. We also rely on the countermeasure of signing the entire protocol

transcript at the end to ensure strong mafia-fraud resistance. Finally to achieve GameTF-security, we add a cheating mode that relies on a trick proposed by Fischlin and Onete [15]. In a nutshell, if the adversary can supply a close-enough guess of the secret key x, the prover will return the correct secret key x as well as the witnesses that were used to generate the commitments of each bit of x that the verifier has. By using these values, the adversary can then authenticate to the verifier.

As future research directions, we mention adding privacy for VSSDB – of which the asymmetric nature of the prover secret key is a necessary first step, as well as investigating the possibility of using other secret-sharing techniques to obtain better distance- and terrorist-fraud resistance.

References

1. Avoine, G., Bingöl, M.A., Kardas, S., Lauradoux, C., Martin, B.: A framework for analyzing RFID distance bounding protocols. Journal of Computer Security 19(2), 289–317 (2011)
2. Avoine, G., Lauradoux, C., Martin, B.: How secret-sharing can defeat terrorist fraud. In: Fourth ACM Conference on Wireless Network Security, WISEC 2011. pp. 145–156. ACM, Hamburg, Germany (June 2011)
3. Bay, A., Boureanu, I., Mitrokotsa, A., Spulber, I., Vaudenay, S.: The Bussard-Bagga and Other Distance-Bounding Protocols under Attacks. In: Information Security and Cryptology - 8th International Conference, Inscrypt 2012. pp. 371–391. Lecture Notes in Computer Science 7763, Springer, Beijing, China (November 2012)
4. Bengio, S., Brassard, G., Desmedt, Y.G., Goutier, C., Quisquater, J.J.: Secure implementation of identification systems. Journal of Cryptology pp. 175–183 (1991)
5. Boureanu, I., Mitrokotsa, A., Vaudenay, S.: Secure and lightweight distance-bounding. In: Lightweight Cryptography for Security and Privacy, pp. 97–113. Springer (2013)
6. Boureanu, I., Mitrokotsa, A., Vaudenay, S.: Towards secure distance bounding. the 20th anniversary annual Fast Software Encryption (FSE 2013) (2013)
7. Brands, S., Chaum, D.: Distance-Bounding Protocols. In: Advances in Cryptology – EUROCRYPT'93. pp. 344–359. Lecture Notes in Computer Science 765, Springer-Verlag, Lofthus, Norway (1993)
8. Brassard, G., Chaum, D., Crépeau, C.: Minimum disclosure proofs of knowledge. Journal of Computer and System Sciences pp. 156–189 (1988)
9. Bussard, L., Bagga, W.: Distance-Bounding Proof of Knowledge to Avoid Real-Time Attacks. In: International Conference on Information Security - SEC 2005. pp. 223–238. Springer Verlag, Chiba, Japan (June 2005)
10. Carbunar, B., Potharaju, R.: You unlocked the Mt. Everest badge on foursquare! Countering location fraud in Geosocial Networks. In: IEEE International Conference on Mobile Ad-Hoc and Sensor, MASS. pp. 182–190. IEEE Computer Society, Las Vegas, NV, USA (October 2012)
11. Desmedt, Y.: Major security problems with the 'unforgeable' (feige)-fiat-shamir proofs of identity and how to overcome them. In: 6th Worldwide Congress on Computer and Communications Security and Protection - SecuriCom '88 (1988)

12. Dürholz, U., Fischlin, M., Kasper, M., Onete, C.: A formal approach to distance bounding RFID protocols. In: Proc. of ISC'11. LNCS, vol. 7001, pp. 47–62. Springer-Verlag (2011)
13. Feldman, P.: A Practical Scheme for Non-interactive Verifiable Secret Sharing. In: Symposium on Foundations of Computer Science - FOCS 1987. pp. 427–437. IEEE Computer Society, Los Angeles, CA, USA (October 1987)
14. Fischlin, M., Onete, C.: Subtle kinks in distance-bounding: an analysis of prominent protocols. In: Proc. WISec'13. pp. 195–206. ACM Press (2013)
15. Fischlin, M., Onete, C.: Terrorism in distance bounding: Modeling terrorist fraud resistance. In: Proceedings of ACNS'13. LNCS, vol. 7954, pp. 414–431. Springer-Verlag (2013)
16. Francillon, A., Danev, B., Čapkun, S.: Relay Attacks on Passive Keyless Entry and Start Systems in Modern Cars. In: Network and Distributed System Security Symposium, NDSS 2011. The Internet Society, San Diego, CA, USA (February 2011)
17. Francis, L., Hancke, G., Mayes, K., Markantonakis, K.: Practical NFC peer-to-peer relay attack using mobile phones. In: International conference on Radio frequency identification: security and privacy issues - RFIDSec'10. pp. 35–49. Springer-Verlag, Istanbul, Turkey (2010)
18. Gambs, S., Onete, C., Robert, J.: Prover Anonymous and Deniable Distance-Bounding Authentication. In: Proceedings of ACM AsiaCCS'14, Accepted for publication. ACM Press (2014)
19. Hancke, G., Kuhn, M.: An RFID distance bounding protocol. In: Security and Privacy for Emerging Areas in Communications Networks, 2005. SecureComm 2005. First International Conference on. pp. 67–73. IEEE (2005)
20. Hermans, J., Peeters, R., Onete, C.: Efficient, secure, private distance bounding without key updates. In: ACM Conference on Security and Privacy in Wireless and Mobile Networks, WISEC'13. pp. 207–218. ACM, Budapest, Hungary (April 2013)
21. Kim, C., Avoine, G., Koeune, F., Standaert, F., Pereira, O.: The swiss-knife RFID distance bounding protocol. In: Information Security and Cryptology–ICISC 2008, pp. 98–115. Springer (2009)
22. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Advances in Cryptology-CRYPTO'91. pp. 129–140. Springer (1992)
23. Ranganathan, A., Tippenhauer, N., Škorić, B., Singelée, D., Čapkun, S.: Design and implementation of a terrorist-fraud resilient distance bounding system. In: Proceedings of the 17th European Symposium on Research in Computer Security (ESORICS'12) (2012)
24. Reid, J., Nieto, J.M.G., Tang, T., Senadji, B.: Detecting relay attacks with timing-based protocols. In: Proceedings of the 2nd ACM symposium on Information, computer and communications security. pp. 204–213. ACM (2007)
25. Tu, Y.J., Piramuthu, S.: Rfid distance bounding protocols. In: First International EURASIP Workshop on RFID Technology, Vienna, Austria (September 2007) (2007)
26. Vaudenay, S.: On modeling terrorist frauds – addressing collusion in distance bounding protocols. In: Proceedings of the 7th Conference on Provable Security (ProvSec'13). LNCS, vol. 8209, pp. 1–20. Springer-Verlag (2013)
27. Vaudenay, S.: Towards Secure Distance Bounding. In: Fast Software Encryption - FSE 2013. LNCS, vol. 7549. Springer-Verlag, Singapore (March 2013), invited talk

A Preliminaries

A.1 General Notations

We give a tabular overview of our notations in Table 2.

p, q	Large prime numbers.
g	A generator of a cyclic group $\mathbb{G}$.
$\mathbb{G}$	Finite cyclic group generated by g, usually $\mathbb{G} = \mathbb{Z}_p^*$.
x	Prover's private key.
$(\mathsf{sk}_{\mathsf{Sign}}, \mathsf{pk}_{\mathsf{Sign}})$	A tuple of private/public secret keys for a signature scheme.
m	The number of fast rounds and bit-length of x.
y	Prover's public key in DBPK-Log.
a_i, b_i, d_i	Commitment values, each for a single bit.
γ_i	Witnesses to open commitments for DBPK-Log.
f	Response function for VSSDB.
γ	Opening function for VSSDB.
c_i	1-bit challenge from the verifier at round i.
r_i	1-bit response from the prover at round i.
k_i, ℓ_i	Randomly-generated bit shares masking x.
Com_i	A commitment to the i-th bit of x in VSSDB.
Cmt	A commitment scheme with algorithms ($\mathsf{Commit}, \mathsf{COpen}$).
SE	A symmetric encryption scheme with algorithms ($\mathcal{E}$, $\mathcal{D}$).
Γ	Discrete logarithm function in DBPK-Log: $\Gamma(x) := g^x$.
Ω	The Pedersen commitment scheme in DBPK-Log.
H	Cryptographic hash function modelled as a random oracle.
H^i	i-th iterative run of the function H.
$\in_R$	Randomly chosen in.
δt_i	RTT for i-th fast challenge-response round.
t_{max}	Proximity bound for the fast-round RTT.
ZKPoK	Zero-knowledge proof of knowledge protocol.

Table 2: Summary of notations.

A.2 The DBPK-Log Protocol

Bussard and Bagga were the first to consider terrorist-fraud resistance in distance bounding [9]. We describe their DBPK-Log scheme in Protocol 4. This protocol is run in a cyclic multiplicative group $\mathbb{G}$ of prime order $p - 1$ generated by an element g. The prover $\mathcal{P}$ has a secret key x along with an associated public key $y := \Gamma(x) = g^x$ certified by a trusted third party and known to the verifier $\mathcal{V}$. DBPK-Log is based on three ingredients: a $(2, 2)$ secret-sharing scheme (i.e., two shares are distributed and both are needed to reconstruct the secret), a homomorphic bit commitment scheme (namely Pedersen's commitment [22]) and a zero-knowledge proof of knowledge (ZKPoK).

Protocol 4: DBPK-Log [9]

Verifier V		**Prover** P
Public key $y := \Gamma(x)$		Private key x
	INITIALIZATION PHASE	
		$k \xleftarrow{\$} \{0,1\}^m$
		$e \leftarrow \mathcal{E}_k(x)$
		For $i =$ to m
		$v_i, w_i \xleftarrow{\$} \{0,1\}^*$
		$a_i = \Omega(k_i, v_i)$
		$b_i = \Omega(e_i, w_i)$
	$\xleftarrow{\{a_i, b_i\}_{i=1}^m}$	
	INTERACTIVE PHASE	
	For $i = 1$ to m	
Choose $c_i \xleftarrow{\$} \{0,1\}$	$\xrightarrow{c_i}$	$r_i = \begin{cases} k_i & \text{if } c_i = 0 \\ e_i & \text{otherwise} \end{cases}$
Measure δt_i	$\xleftarrow{r_i}$	
	VERIFICATION PHASE	
	$\xleftarrow{\gamma_i}$	$\gamma_i = \begin{cases} v_i & \text{if } c_i = 0 \\ w_i & \text{otherwise} \end{cases}$
	$\xleftrightarrow{\mathsf{ZKPoK}[\text{well formed}]}$	

Accept if and only if all r_i are coherent with respect to a_i, b_i, γ_i, $\delta t_i \leq t_{\mathsf{max}}$ $\forall i$, and ZKPoK verifies.

PROTOCOL DESCRIPTION. During the *initialization phase* the prover picks a random m-bit integer k and uses it to encrypt the long-term secret x as $e \leftarrow \mathcal{E}_k(x) = x - k \bmod p$. The values k and e act as shares in a (2,2)-secret sharing scheme. Then, the prover commits in a bitwise manner to each bit of each share using a homomorphic commitment scheme[1], which leads to commitments of the form $a = \{a_i\}_{i=1}^m$ and $b = \{b_i\}_{i=1}^m$, with $a_i = \Omega(k_i, v_i) = g^{k_i} h^{v_i} \bmod p$ and $b_i = \Omega(e_i, w_i) = g^{e_i} h^{w_i} \bmod p$. Finally, this string of commitments is sent to the verifier, thus completing the initialization phase.

Afterwards during the *interactive phase*, the prover and verifier exchange binary challenges and their corresponding binary responses to estimate the roundtrip-time (RTT). Each response is equal to either a bit of the random share k or a bit of the encrypted secret e.

Finally during the *verification phase*, the prover sends the randomness (either v_i or w_i) used to generate the commitment of the fast round responses (this value is denoted as γ_i). Since the commitment is homomorphic, the verifier can compute $z = \prod_{i=1}^m (a_i b_i)^{2^{i-1}} = \Omega((k+e), v)$. Then, the prover and verifier runs a ZKPoK protocol in which the prover demonstrates the knowledge of a tuple (x, v) such that $z = \Omega(x, v)$ and $y = \Gamma(x)]$. The verifier accepts this proof if (1) all the fast responses are coherent with respect to the committed values, (2) all RTT values are below t_{max} and (3) the ZKPoK succeeds.

[1] The authors suggest Pedersen's commitments [22].

A.3 Attacks on DBPK-Log

Avoine, Lauradoux and Martin [2] proposed a key-recovery attack against DBPK-Log as well as other DB protocols. This key-leakage attack also enables to conduct a mafia fraud against another protocol due to Bussard and Bagga (see [14] for more details). However, this key-recovery attack does not work if the prover is honest in the DBPK-Log protocol. Indeed when sent the complement of the challenge, the prover will also send the randomness to open the commitment for the complementary challenge, thus preventing the adversary from knowing whether or not $k_i = e_i$ for some round i.

However at Inscrypt 2012, Bay and co-authors [3] proposed another terrorist fraud whose probability of success is $\frac{1}{2}$. Their attack is based on the observation that the homomorphic commitment check proceeds over *all* the commitments. Thus, the adversary can generate random values for the shares k and e and use them, as long as it sacrifices one of the commitments to add a value given by the prover ensuring the correct homomorphic verification. More precisely, the prover computes $z' := \Omega(x, v')$ with a newly generated v' and sends z' to the adversary $\mathcal{A}$. Now $\mathcal{A}$ generates two random m-bit strings k and e, for which he generates honestly the commitments a_i and b_i (for $i = 1$ to m).

Using these values, $\mathcal{A}$ constructs two strings of commitments to send to the verifier, using as many of the honestly-computed commitments as possible, but also ensuring that the homomorphic verification still holds. More specifically $\mathcal{A}$ sends two arrays A and B with m elements each, which he fills from position 2 to m with the commitments a_i and b_i that he has generated himself. For A_1 and B_1, the adversary guesses c_1 with probability $\frac{1}{2}$ and inserts in the corresponding array (A if $c_i = 0$ and B otherwise) the correct, self-generated commitment (i.e., respectively a_1 and b_i). In the other array (corresponding to the complement of c_1), $\mathcal{A}$ adds a value ensuring that $z = \prod_{i=1}^{m} (A_i B_i)^{2^{i-1}} \bmod p$.

Now $\mathcal{A}$ can correctly answer to all challenges and reveal unconditionally the correct randomness for the rounds 2 to m as well as the correct randomness for the round 1 if he has correctly guessed c_1. Afterwards, the malicious prover helps the adversary to succeed in conducting the ZKPoK Thus, no information about x is leaked (except a commitment and a ZKPoK, which reveals nothing) and the adversary $\mathcal{A}$ wins with probability $\frac{1}{2}$.

B Security Proofs of VSSDB

In this section, we present the detailed security proofs of the VSSDB protocol (Section 3.1).

Proof. Distance-fraud resistance. Our proof relies on the fact that for each mode $\hat{\mathsf{M}}_i$, the XOR of the responses for the two challenges $\mathsf{f}(\hat{\mathsf{M}}_i, 0) \oplus \mathsf{f}(\hat{\mathsf{M}}_i, 1) = x_i$, whose value is 1 with probability exactly $\frac{1}{2}$ if x is chosen at random in an honest manner. In particular, for each mode the adversary has a probability of at most $\frac{3}{4}$ to win the round, regardless of his choice of k_i and ℓ_i.

We first rule out the possibility that the adversary cheats on the commitments he sends in $\mathsf{c_P}$. Consider an adversary that, concretely, chooses a value $k_i = 0$ (without loss of generality) in the initial phase of one of the q_V sessions, computes a_i for this value using some witness u_i, and sends it within $\mathsf{c_P}$. Later, the same adversary is queried on k_i, sends $\tilde{k_i} = 1$, and yet manages to authenticate. For this adversary, we can easily build a reduction $\mathcal{A}^*$ that wins the binding game against the commitment scheme. The same holds for a commitment on a value ℓ_i for a specific i, and for a commitment on a value e_i (once we assume that the output of the function H is truly random, for which we lose a total of $mq_V\mathbf{Adv}^{\mathsf{H-PRF}}_{\mathcal{A}_1}$). Thus, the probability that the adversary, having committed to the values k_i, ℓ_i, e_i, is able to respond with a different value when queried in the time-critical phases is: $3mq_V\mathbf{Adv}^{\mathsf{Com.Bind}}_{\mathcal{A}_2}$ (accounting for the length of each string, and for q_V sessions against the verifier).

Now consider an adversary who is able to cheat on the generation of the value e_i (i.e., an adversary who can generate e_i such that $e_i \neq \mathcal{E}_{k_i||\ell_i}(x_i)$). Assuming the correctness of the decryption process (losing a term $q_V m\mathbf{Adv}^{\mathcal{E}().\mathsf{Corr}}_{\mathcal{A}_3}$) and of the homomorphic commitments respectively (losing a term $q_V m\mathbf{Adv}^{\mathsf{Com.Corr}}_{\mathcal{A}_4}$), this leaves the possibility that the adversary was able to open the (bitwise) commitment of x to a different value $x' \neq x$. More formally, the adversary was able to find some witnesses $\{\hat{\nu_1}, \ldots, \hat{\nu_m}\}$ such that at least one witness $\hat{\nu_i} \neq \nu_i$, which ensure $\mathsf{Com}_i = \mathsf{Commit}(\nu_i, x_i) = \mathsf{Commit}(\hat{\nu_i}, x'_i)$. However, since the witnesses must be used to generate the NIZK proof π sent together with $\mathsf{c_P}$, if the latter is sound (we lose a term $q_V\mathbf{Adv}^{\mathsf{NIZK.Sound}}_{\mathcal{A}_5}$), then we can extract the witness that breaks the binding property of the commitment. The latter can only happen with probability $mq_V\mathbf{Adv}^{\mathsf{Com.Bind}}_{\mathcal{A}_6}$. As discussed before, the remaining probability to win is now upper-bounded by a probability of at most $\frac{3}{4}$ of winning a round for each of the two modes, thus leading to a global probability of $\frac{3}{4}$ per round per verifier-adversary session.

Mafia-fraud resistance (without cheat modes). For mafia-fraud resistance, we first rule out the possibility that $\mathcal{A}$ sends a ciphertext $\mathsf{c_P}$ that the prover has not generated before. In particular, we consider a slightly modified game, in which we keep track of values $\mathsf{c_P}$ that the adversary has seen or extracted from the prover in the at most q_{OBS} prover-verifier and the at most q_P adversary-prover sessions. In this game, the verifier immediately aborts if the adversary sends (in a verifier-adversary session) a ciphertext $\mathsf{c'_P}$ different from all the observed/extracted ciphertexts. These two games are equivalent as long as the verifier detects any unobserved/unextracted ciphertext injected by the adversary $\mathcal{A}$. If the decryption of the public-key encryption scheme is correct (we lose a term $q_V\mathbf{Adv}^{\mathsf{PKE.Corr}}_{\mathcal{A}_1}$) and the signature scheme is complete (we lose a term $q_V\mathbf{Adv}^{\mathsf{Sign.Corr}}_{\mathcal{A}_2}$), we can build a reduction between the equivalence of the two games and the unforgeability of the signature scheme. This reduction outputs the string parsed as the signature in the decryption of $\mathsf{c'_P}$ as the forgery for the string parsed as the message m_P in the decryption of $\mathsf{c'_P}$. Such a possibility can be ruled out, except with probability $q_V\mathbf{Adv}^{\mathsf{Unf}}_{\mathcal{A}_3}$.

Thus, the adversary cannot inject self-generated commitments into a verifier-adversary session. In addition, as in this game the prover is honest, the values for k_i,ℓ_i, and e_i (due to the choice of x) are all 0 with probability. $\frac{1}{2}$ for each session. Furthermore, for each given mode, the probability that the two responses are equal is $\frac{1}{2}$.

Since the public-key encryption scheme is IND-CCA, the adversary cannot distinguish between two ciphertexts $\mathsf{c_P}$ generated for the same auxiliary values k and ℓ, or for different values, except with total probability $(q_{\text{OBS}} + q_P)\mathbf{Adv}^{\mathsf{PKE.IND-CCA}}_{\mathcal{A}_4}$. Similarly, due to the zero-knowledge property of the NIZK proof of knowledge, the adversary is unable to tell whether two proofs π and π' are generated using the same witness (we lose here a term $(q_{\text{OBS}}+q_P)\mathbf{Adv}^{\mathsf{NIZK.ZK}}_{\mathcal{A}_5}$).

In particular, the only way an adversary can *know* whether two sessions share the same responses is to forward the encryption $\mathsf{c_P}$ and the proof π) from an adversary-prover session sid' to a verifier-adversary session sid (these two sessions share the answer). Moreover, we rule out the possibility of a replay of the ciphertext by the adversary due to the fact that two verifier-adversary sessions will share the nonce N_V with probability $\binom{q_V}{2}2^{-|N_\mathsf{V}|}$. Thus, we now have that any verifier-adversary session sid shares responses with at most one adversary-prover session sid'.

We now examine the adversary's strategy for the interactive part of the protocol, considering in particular a single time-critical round i. The possible situations are the following.

- Sessions sid and sid' share the same mode M_i (with probability $\frac{1}{2}$). In this case, the adversary maximizes his probability to win this phase if he guesses the *challenge* or the *response* (both happen with probability $\frac{1}{2}$). Else, if he wrongly guesses the challenge (or sends the wrong challenge/response on purpose), then he fails to authenticate, except with probability $\mathbf{Adv}^{\mathsf{Unf}}_{\mathcal{A}}$. This probability corresponds to the possibility of building a reduction against the unforgeability of the signature scheme (specifically the final signature in the protocol).
- Sessions sid do not share the same mode M_i for this phase. In this case, the best strategy of the adversary is to guess the response (since guessing the challenge will not help), which happens with probability $\frac{1}{2}$. However in this case the adversary will not be able to use the signature given by the prover in session sid' since the message signed includes the value of the modes.

Overall, regardless of whether the two sessions share or not the same mode, the adversary has a probability of winning each round with an independent probability of $\frac{1}{2}$ per time-critical phase (to a total of $q_V 2^{-m}$).

Mafia fraud (with cheating). We also have to account for an adversary that can guess the key x sufficiently close to be given x and $\mathsf{sk}_{\mathsf{Sign}}$ by the prover. Similarly to the proof of Fischlin and Onete [15], we sum up over all the "near guesses" (with the respective probabilities of having the correct responses), which leads to a global probability of $q_P 2^{(-2-\log_2 3)m}$.

Impersonation resistance. Similarly to the proof for mafia-fraud resistance, we first rule out the possibility that two sessions share responses (unless they

are a verifier-adversary session and an adversary-prover session run at the same time). Thus, any verifier-adversary session the adversary opens has at most one adversary-prover session with which it shares respectively modes and commitments. At this point, the definition requires that the adversary wins without producing the same transcript as the concurrent adversary-prover session. We upper-bound this probability by observing that any change in the transcript requires the adversary to be able to forge the signature, thus obtaining the claimed bound. This bound also relies on the fact that, for the commitments we have chosen, it is not possible to find two witnesses w and w' such that $\mathsf{Commit}(x, w) = \mathsf{Commit}(x, w')$. This latest issue is considered separately, since in the verification phase the witnesses are not signed. We furthermore integrate, as for mafia fraud, for the probability of successfully receiving x and $\mathsf{sk}_{\mathsf{Sign}}$ from the prover.

GameTF. This statement is proven in two steps. We first consider a successful terrorist fraud adversary $\mathcal{A}$. Our goal is to show that either this adversary has a negligible success probability or he is helpful to a mafia-fraud adversary $\mathcal{A}'$ in the sense of [15]. In the first step of the proof, we show how the adversary $\mathcal{A}'$ reconstructs a guess of the secret x by running $\mathcal{A}$ as a black box. Afterwards, we show that this gives $\mathcal{A}'$ a non-negligible probability to recover x and $\mathsf{sk}_{\mathsf{Sign}}$, and thus trivially authenticate afterwards.

We consider an adversary $\mathcal{A}'$ who runs $\mathcal{A}$ internally, and who ends up with a guess $\hat{x}$ of x. For each verifier-adversary session of the terrorist-fraud adversary $\mathcal{A}$, the adversary $\mathcal{A}'$ queries $\mathcal{A}$ for each fast round and for each of the two challenges ($\mathcal{A}'$ does not change the mode). If $\mathcal{A}$ answers on both branches, $\mathcal{A}'$ sets $\hat{x}_i$ to be the XOR of the two responses. If the adversary does not respond on at least one branch, $\mathcal{A}'$ sets $\hat{x}_i$ to be a randomly chosen bit. This strategy yields the following alternatives.

- $\mathcal{A}$ knows both bits correctly. In this case, $\mathcal{A}$ always passes the round, but the guessed bit $\hat{x}_i$ is also correct.
- $\mathcal{A}$ gets exactly one bit correctly. In this case, $\mathcal{A}'$ has the wrong bit. On the other hand, the adversary fails the session with probability $\frac{1}{2}$.
- $\mathcal{A}$ gets both bits wrong, in which case $\mathcal{A}$ fails the session, but $\mathcal{A}'$ gets the right bit.
- $\mathcal{A}$ refuses to answer at least for one branch. At this point, the adversary has a probability of at least $\frac{1}{2}$ to fail, while $\mathcal{A}'$ guesses the corresponding guessed bit of x is $\frac{1}{2}$.

As outlined in the similar proof in [15], the adversary $\mathcal{A}'$ will recover as many bits of x as is the first adversary's ($\mathcal{A}$'s) success probability to pass the fast rounds. With this probability, $\mathcal{A}'$ now recovers x and $\mathsf{sk}_{\mathsf{Sign}}$ by using the cheating mode. Thus, if we assume that $\mathcal{A}$ wins with a non-negligible probability, so does $\mathcal{A}'$ in the subsequent attack. In particular, $\mathcal{A}$ has been helpful to $\mathcal{A}'$ since the protocol is mafia-fraud resistant.

□